第29卷 第7期
2006年7月
计 算 机 学 报
CHINESE JOURNAL OF COMPUTERS
Vol. 29 No. 7
July 2006

会话类E-Service的接口兼容和服务组合分析

张文涛 彭 泳 陈俊亮
(北京邮电大学网络与交换技术国家重点实验室 北京 100876)

摘 要 针对会话类E-Service的会话完整性对环境提出的更多限制以及由此导致的行为差异，给出了对接口环境的严格定义和接口兼容检查的精确算法. 在应用上，由于接口兼容的组合算法COMP可以得到组合接口自动机，通过计算 $COMP(COMP(A_1,\cdots,COMP(A_{N-1},A_N)))$ 可以保证最终的组合结果是协议兼容的.

关键词 接口兼容；组合；会话类E-Service；有限状态自动机(FSM)；Web服务
中图法分类号 TP311

Interface Compatibility and Composition of Session-Oriented E-Service

ZHANG Wen-Tao PENG Yong CHEN Jun-Liang
(*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing* 100876)

Abstract Session-oriented Electronic Service(E-Service)'s observable behavior presents a defined choreography of messages, characterized in terms of temporal and logical dependencies among the exchanged messages, as well as a "start" and an "end" in the message sequence. An automata based formalism interface model is presented to capture the temporal aspect of message flow. The formalism supports automatic verification of interface compatibility which can be regard as type system for interface interaction. The interface compatibility check focuses on the more constraint that interfaces will put on environment due to integration of E-Service, and the strict definition of environment behavior and the precise algorithm named COMP. The COMP outputs a composite interface model which can be used to assist(dynamic) composer to ensure the correctness of composition in term of protocol compatibility.

Keywords interface compatibility; composition; session-oriented E-Service; finite states machine; Web Services (WS)

1 引 言

SOC(Service Oriented Computing)以服务为基本构件，通过SOA框架研究服务的发布、协调和交互，将来自Internet上不同提供商的服务，通过对相关功能的组合，得到更加丰富而且具有增值效果的组合服务，并以此为基础构建新的应用. 这种方式改变了传统的业务提供和部署模式，因此引起了人们极大的兴趣.

服务组合首先要有好的服务描述模型来充分地表达服务的语义和交互行为，这样才能提供精确的服务查找和匹配，并确保服务之间正确交互和组合行为的验证. 通过对现实E-Service应用的调查，文献[3]认为应该把E-Service分为离散的服务系统和交互式的服务系统(也就是会话类E-Service)两类. 会话类E-Service一般是一个长期的需要状态信息的过程，其服务交互表现为时序性和完整性两个

收稿日期：2006-01-11；修改稿收到日期：2006-05-15. 本课题得到国家自然科学基金(60432010)资助. 张文涛，男，1979年生，博士研究生，研究方向为E-Service的描述、(动态)组合、协议验证. E-mail：taoge97@bj1860.net. 彭 泳，男，1978年生，博士，讲师，研究方向为网络服务、移动网络. 陈俊亮，男，1933年生，博士生导师，中国科学院与中国工程院院士，研究领域为通信网络、通信软件.

特征,方法的调用依赖于当前状态,且交互是一个完整的过程.传统的类似于 WSDL[8] 的静态模型不能表达服务的过程和状态信息,因此,无法描述会话类 E-Service的动态接口特征.而新出现的 BPEL,WSCI 等规范通过基于状态的模型来描述单个服务的控制流和状态,告诉调用方如何与服务进行交互,为会话类 E-Service 动态的接口描述提供了比较好方法.

会话类 E-Service 的动态接口特征对服务的组合技术的影响表现在:服务的查找不能仅仅检查单个服务方法的输入输出或前置后置条件是否能满足条件,必须要全局考虑方法的调用在服务中需要满足什么样的时序关系以及服务之间的交互是否一致;服务组合也不仅仅是功能的罗列与集合,还需要考虑原子服务之间的行为交互.组合内部全体服务的交互一致性称之为协议兼容;而某两个服务之间的行为接口一致性匹配则称之为接口兼容.我们的目的是通过对会话类 E-Service 接口兼容的研究,提出一种组合的 Design-Time 验证方法,来保证最后得到的组合服务是协议兼容的.

本文参考现有的服务行为接口规范(如 WSCI[7],BPEL[1]等),提出了一个会话类 E-Service 行为接口的轻量级形式化模型,以此检查 E-Service 的接口兼容,并应用于服务组合框架中,对组合过程进行(Design-Time)验证,形成基于动态行为的类型系统.这种轻量形式化主要定义服务交互过程中消息流(映射为服务方法调用与返回)的逻辑和时序关系(也称为协议规范),而接口兼容研究主要针对会话类 E-Service 的会话完整性特征和对服务接口环境的严格定义,由此得到接口交互行为和某些特殊状态集合的精确定义,这样为算法的正确性和精确性提供了保障.接口兼容检查的 COMP 算法的输出是接口对的组合自动机.组合自动机完全表达了接口对可能的兼容行为,这样可以保证通过 COMP 检查得到的最终组合肯定是兼容的.

本文第 2 节分析相关工作以及本文的贡献;第 3 节对服务交互和组合进行讨论,提出本文要解决的问题;第 4 节以 WSCI 为基础定义 E-Service 动态接口的形式化模型;第 5 节先简单地讨论基本的协议兼容,然后重点讨论接口兼容的定义、判定方法以及合法的组合接口自动机的计算方法;最后结合实例验证算法,并做结论.

2 相关工作

文献[12]对不同的 Web Service 之间存在的功能依赖进行建模,作者将服务方法建模为单位功能,描述功能的单向组合,并从多个层次定义子功能的组合和时序关系对组合功能的影响.但是较少从整体上考虑服务模型,孤立了服务内部方法的联系,缺乏对方法调用之间的逻辑和时序关系的描述.文献[11]在 Semantic Web Service[5](OWL-S[16]等)描述的基础上添加了组合规范(compositional specification),用 assumption 和 commitment(A-C 范式)来定义业务在执行过程中的状态变化,扩展了 OWL-S 对 E-Service 的输入输出和 initial,final 状态的定义,并以此为基础进行服务组合的推理和验证.文献[10,20～22]等对服务组合的全局异步行为进行研究,对协议整体行为进行分析,是自顶向下的方法.

软件接口模型和接口兼容问题在软件组件领域已有所研究.文献[13]简化了接口交互,完全不考虑其它实体对接口交互的影响,很多情况下不适用.文献[9]提出了接口自动机,研究组件的接口兼容问题,通过分析环境对接口兼容的影响,提出了不同于 I/O 自动机的“乐观”方法.但是它未定义终止状态,因此不能描述会话类 E-Service 的完整性,无法解决会话类 E-Service 的接口兼容检查.另外,文献[23]没有严格定义环境自动机的行为,导致最后的接口兼容检查算法中没有体现出接口自动机应该具有的行为特征,比如某些非法状态不能被检测出来,接口可执行的动作没有精确定义等,而这些都是算法的关键元素,会直接影响算法的正确性.

我们工作的贡献在于:相比于文献[12]的单向功能组合,我们用具有输入输出的 FSM(米兰机)来描述会话类E-Service 的行为接口.这种描述更倾向于交互行为,从整体上兼顾方法之间的依赖和内部方法与外部环境的相互依赖(交互),而不只是讨论单个方法对外部环境的单向功能依赖.因此,其对接口的描述更加全面,更加适合会话类 E-Service.

其次,在自动机中严格定义终止状态来表示 E-Service 的会话完整性.这样死锁成为兼容定义必须要考虑的问题.相比于文献[9]的接口自动机,完整性的存在会带来两方面的影响:(1)接口对环境的限制更多;(2)非法状态的定义需要扩充.针对这两个问题,文章对接口兼容进行重新定义和分析,特别是对环境的定义,由此可以得到新的非法状态集合和接口可执行动作集合.

然后,严格地定义了和刻画环境自动机的行为.乐观方法的思想是通过环境的定制改变对接口交互

的行为，其最核心的内容是组合自动机行为和非法状态集合的精确定义，它们直接关系到算法的正确性和精确性.

最后，给出检查接口兼容的算法 COMP. 由算法生成的组合接口自动机，作为两个接口行为的组合，可以用于进一步的接口兼容检查. 依此类推，COMP 算法可应用于组合过程，为服务组合提供设计时的检查和验证，自底向上的组合服务.

3 服务交互和组合

现实中服务的交互依赖一系列的标准和规范，如服务方法的调用是采用 SOAP 协议的请求和应答，以 XML 格式封装消息；服务的接口定义是基于 XML 格式的 WSDL[8] 描述，并通过 XML 的 schema 定义服务接口的消息类型.

服务组合就是将一系列功能相关的"原子"服务组合成为一个新的服务，提供更加完整的功能. 服务的组合可以分为两种：(1)功能序列组合. 其特点是对 N 个服务功能进行简单罗列，服务之间没有交互；(2)交互组合. 服务之间通过消息交互各自完成服务会话从而得到完整的组合. 会话类 E-Service 的组合属于交互式组合，要求组合的原子服务不仅仅是功能的组合、服务方法的 IOPE 匹配，而且要保证服务交互是一致的.

旅行代理(类似于文献[7]中的例子)为客户提供旅行计划、订票等服务，一共有三个实体：旅行代理服务、旅行者(客户)、飞机场等. 旅行代理根据客户提供的旅行信息，为客户提供旅行计划；客户可以修改、取消计划，也可以向旅行代理进一步确认；经过确认后，旅行代理向机场预定机票，此时客户可以获得实际的机票，也可以取消机票预定.

旅行例子中的三个服务实体之间的交互满足一定的规程(称为协议)，其中每个实体对消息的处理也各有自己的逻辑，对应协议中和自己相关的部分交互. 如果实体都遵守其行为规范，且所有实体的交互是一致的，那么就可以认为这些服务是协议兼容的. 我们感兴趣的是如何通过检查任意两个实体之间的行为是否一致(即接口兼容检查)，来保证最后得到的组合也是协议兼容的. 这个逐步的兼容检查过程是和实际的组合服务设计和开发过程一致的，即逐个地查找功能匹配且行为一致的服务，从而达到组合的目的. 图 1 是接口兼容检查在组合模型中的位置.

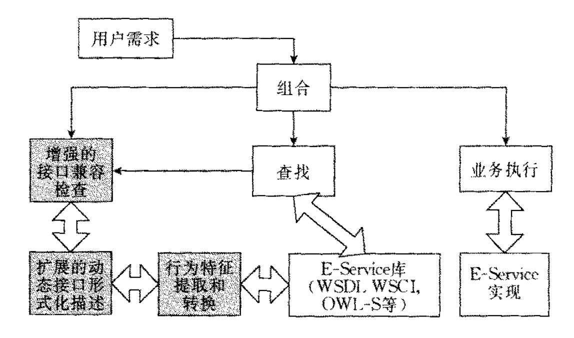


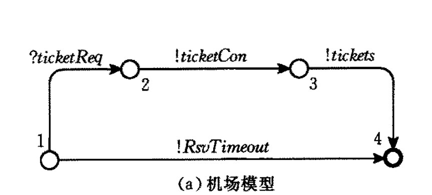
图 1　基于接口兼容检查的组合模型

4 E-Service 接口模型

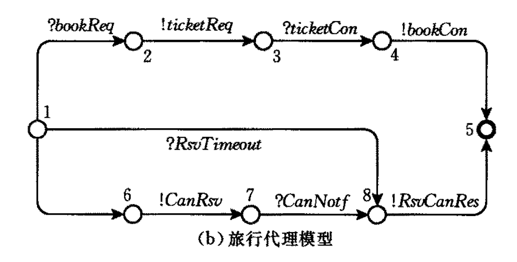
本节定义 E-Service 行为接口的形式化模型，通过这个模型定义 E-Service 接口的动态行为，即消息交互的时序和逻辑关系，并以此作为进一步分析的基础.

本文的会话类 E-Service 的形式化模型充分考虑了现有的工作和技术规范，如 WSCI 的 choreography 接口描述模型、BPEL 的抽象模型等，同时考虑到现有 E-Service 的发展(如在电信领域的 Parlay/OSA，3GPP，OMA 等)使会话类的 E-Service 越来越受到重视. 我们选择带有输入输出的有限状态自动机 FSM(米兰机[4])作为对会话类 E-Service 的建模工具. FSM 在响应式系统[19]、通信协议[2]和组件接口[9,13]的研究中已有广泛的应用，而且 WSCI 和 BPEL 语言的核心模型也是以 FSM 为基础的.

图 2 的接口 FSM 模型是旅游代理例子，我们对



(a) 机场模型



(b) 旅行代理模型

图 2　简化的旅行代理业务的 FSM 模型

旅游代理服务和机场服务两个接口进行FSM自动机建模.为了便于讨论,我做了相应的简化:只取了预约订票结束后的情景,机场逻辑也不允许取消订票.圆圈表示状态,弧线代表状态转移及消息的传递,消息的接收和发送分别以问号(?)和感叹号(!)表示.

现在我们来形式化地定义E-Service的接口模型.服务的接口模型是一个带有输入输出的有限状态自动机(简称为接口FSM模型) $E=\{S, sinit, F, T, N, [!/?]M_{i=1\cdots N}\}$,其中 S 代表E-Service的有限状态集合,$sinit \in S$ 是初始状态,代表服务会话的开始,N 代表与服务有直接消息交互关系的其它服务(称为交互对象)数量,M 是服务的消息集合,可以再细分为 $[!/?]M_i$,表示同标识为 i 的交互对象之间的发送或接收消息集合(省略方向标记则表示两者之和),$F \subset S$ 是E-Service的终止状态集合,T: $S \times [!/?]M_i \rightarrow S$ 是状态转移函数,服务接收或发送一个消息将导致状态转移,服务在一个状态中只能发生一个动作[10]. $T(s,[!/?]m)$ 可以表示实体在状态 s 时处理完消息 m 后进入的状态.

在接口FSM中,service可能收到在当前状态不可接收的消息,这种消息对service来说称为不期望的消息.不期望的消息将导致交互错误,意味着消息交互行为不一致.Service也可能永远无法接收到期望消息,导致session无法完成,这也是不一致的现象.

下面对FSM(有限状态自动机)表示的服务接口(称为接口FSM)的语义进行一些补充说明.接口FSM不仅规定了service自身的行为规则,同时也对其它与之交互的E-Service实体(称为环境)的行为进行了限制.这种限制的依据是service本身的状态,体现在三个方面:环境不能发送service不期望的消息,环境必须能够接收service的发送消息,而且环境不能使service停止于任何非终止状态.相比于文献[9],我们对环境提出了更多限制,这是会话类E-Service的一个重要特征.因为会话类E-Service需要完整性,任何停止于非终止状态都是属于死锁的范畴,因此要求环境总是尽量去满足service的执行条件,即在任何一个状态,环境要保证至少有一个动作是可以执行的.在第4节我们将详细讨论如何检查满足会话完整性的接口兼容.

本文讨论的消息交互限制在同步环境[13],不考虑消息通信时延、队列缓存和处理时间.文献[20]提出了三个同步条件.满足同步条件的协议在异步环境下的行为和在同步环境下一样,也是可以判定为兼容的.在同步环境中,只有同时存在发送消息 m 的实体和接受消息 m 的实体,消息交互才是可以执行的;否则可以判定为未指定的接收[2]或不可执行接收[2].

后面文中使用的标识定义如下:

大写字母 A,B,C 代表实体,下标表示实体变量,如 A_i 表示实体 i,S_i 代表实体 i 的状态集合,s_i 属于 S_i;M_j 表示与实体 j 交互的消息集合.小写字母 s,t,u 等表示状态;小写字母 a,b,c 表示消息,某些特殊的消息用大写字母 E,N,M 等标识.

5 接口兼容

5.1 概 述

对于 N 个实体组合的协议兼容可以容易地判定[13].然而在业务组合过程[5,15,17,18]中,一个完整的组合是通过一个个地添加满足目标功能的实体而形成的,如果等所有的组件都添加完毕再检查协议是否兼容,那么需要检查的目标组合的数量级将会非常巨大.因此,需要引入接口兼容检查,在组合的每一步过程中验证当前选择添加到组合中的实体能否和已存在的实体集合在行为上兼容,以此来保证最后获得的完整组合必定是兼容的.

接口兼容的分析方法不同于协议[2,13],对于协议兼容,只要协议中存在某个状态的发送消息没有相应的接收者(未指定接收[2]),那么就是非法状态.但是这个判定条件在接口兼容不能成立,首先接口兼容针对的是不完整的组合,对于接口交互双方而言,还存在与外部实体的交互,虽然这部分交互没有明确地指定接收者,但是不能定义为未指定的接收.为此,将所有可能的外部实体统一用环境代表(称为环境自动机),环境是这些外部实体的消息理所当然的接收者,且环境和接口对三者组成一个完整的组合(简称 *Penv*).另外我们知道接口自动机模型不仅规定了服务自身的行为规则,同时也对环境行为进行了限制,这种限制的依据是接口自动机的状态,内容体现在三个方面:环境不能发送服务不期望的消息,环境必须能够接收service的发送消息,而且环境不能使服务停止于任何非终止状态.在满足上面这三个条件的情况下,环境的行为可以任意定制.满足接口限制的环境定制,不会影响接口和环境交互的兼容性,但是会影响接口的行为,比如,使某些状态永远执行不到而成为不可达状态.那么适当地调

整环境的行为，就可以使组合 *Penv* 中可能出现的某些非法状态(如接口对之间存在的未指定接收)成为不可达状态. 如果存在一个环境满足接口 *A* 和 *B* 的限制，并且环境和接口对组成的组合是协议兼容的，则可以认为这两个接口就是接口兼容的. 这就是所谓的"乐观"方式，由此得到接口兼容的定义和充要条件.

定义 1(充分必要条件).　*A*，*B* 接口的交互是接口兼容的，当且仅当存在一个合法环境 *ENV* 能保证接口交互与环境组成的协议 *Penv*(*A*，*B*，*ENV*) 的可达状态集合中没有非法状态，且每个可达状态都能通过有限步执行到达终止状态.

接口兼容的定义中要求每个可达状态都能通过有限步执行到达终止状态，是为了满足会话类 E-Service 的会话完整性，防止服务交互的时候出现死锁和活锁等情况.

图 3 是对图 2 中简化了的旅游者、旅游代理和飞机票场景分别用两个方法对其进行接口分析的例子，为了便于讨论去掉了部分无关的自动机内容. 如果直接对图 2 旅游代理和飞机票代理进行组合，会发现存在非法状态，即旅游代理在状态 1 接收到取消定票请求 *rsvCanReq* 时，会向飞机票代理发送取消请求 *CanRsv*，而飞机票代理此时所处的状态 1 却没有相应的取消功能，即当前状态不能接收消息 *CanRsv*，则导致未指定的接收动作的产生. 图 3(a)所示的是旅游代理和飞机票代理的组合接口模型(简称组合模型或者接口组合)，虚线标注可以到达终止状态的执行路径，实线标注存在非法状态的执行路径，由于 *Penv* 存在非法状态因此不能满足充要条件，得出结果是接口不兼容. 而乐观的方法则只需要存在一个环境能让整个协议兼容，那么就认为两个接口模型是兼容的. 如图 3(b)提供的旅游者接口模型作为环境，旅游者的状态 1 不存在取消定票请求，因此旅游代理在状态 1 不会接收到 *CanRsv*，使非法状态 8 成为不可达状态，从而得到组合模型图 3(c). 图 3(c)相比于图 3(a)，没有了那条虚线标注的路径，这样使得非法状态成为不可达状态，而剩下的两个实线标注的路径中，所有的状态都能在有限步内到达终止状态，即没有死锁或不可终止的状态. 这样图 3(c)的 *Penv* 模型是满足兼容的充要条件的，由此可以得出接口兼容的结果.

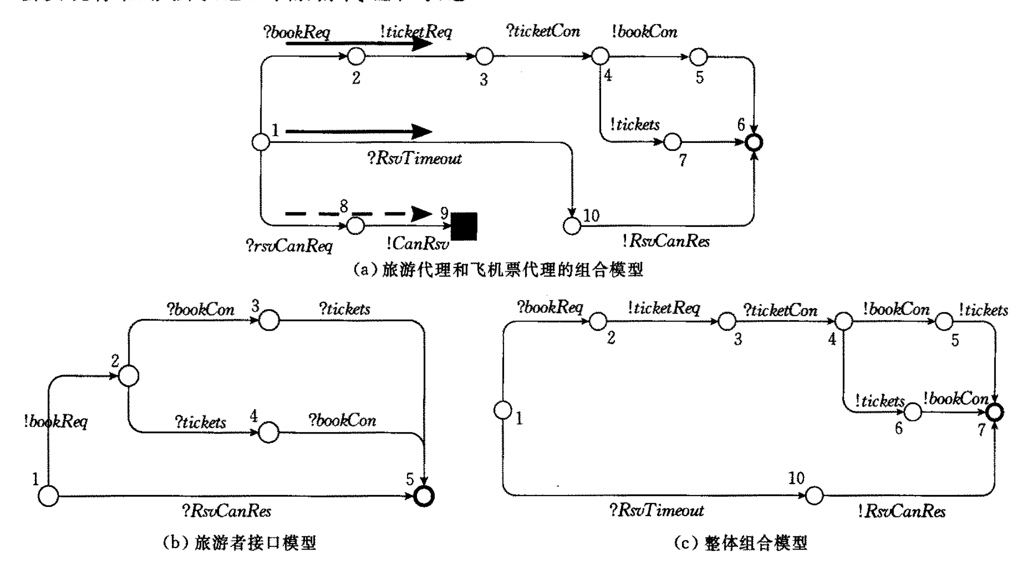


图 3　环境对接口兼容的影响

5.2　接口兼容

下面从环境自动机的定义入手，分析协议自动机 *Penv* 的行为特性，并根据接口兼容的充要条件，逐步地修剪 *Penv* 使之最终得到一个满足接口兼容定义的 *Penv*′，由此可以判断接口的兼容性.

定义 2.　协议 *Penv* 是由接口 *A*，*B* 和环境 *ENV* 组成的有限状态自动机，其中状态集合 $SG \subset S_A \times S_B \times S_{ENV}$，初始状态 $(sinit_A, sinit_B, sinit_{ENV})$，全局转移 $TG: SG \times M = SG$，全局状态集合 *SG* 由初始状开始通过 *TG* 遍历得到，终止状态 $FG = F_A \times F_B \times F_{ENV}$，且 $FG \subset SG$.

对 *TG* 的具体定义，即任何一个状态下，到底哪些 *TG* 是属于协议 *Penv* 的，则还要看环境自动机的定义. 我们知道，对于接口兼容中的环境，需要分别

满足两个接口的条件限制，但是当接口之间存在交互的时候，问题就变得复杂了．因为接口之间交互导致接口状态变化，而这个变化环境无法感知，也就是说环境的状态不会变化，那么环境必须要满足接口之间在内部交互前后的状态的限制，也就是说环境的某一个状态有可能需要满足接口在不同状态的限制．而当这几个状态对环境的限制条件存在冲突时，环境就根本无法满足接口交互需要，因此此时不存在任何环境能满足接口交互的条件限制，这时可以说明在当前状态下，接口之间必然存在行为不一致．我们定义满足接口自动机限制条件的环境为合法环境，任何一个单个自动机的合法环境必定能保证环境和接口协议兼容，而接口对的合法环境只能保证环境不会产生非法输入和必须接受接口组合的合法输出．

定义 3. 如果环境能满足接口兼容中两个接口的限制条件，则称环境是合法的．接口兼容的合法环境需要满足以下条件：

对协议 $Penv(A,B,ENV)$ 中的任一可达全局状态 $g(u,v,senv)$，

(1) 如果状态 g 中存在内部交互和接收消息集合 $?E=\{\varepsilon_1,\varepsilon_2,\cdots,\varepsilon_N\}$ 发送消息集合 $!E=\{\varphi_1,\varphi_2,\cdots,\varphi_N\}$，不失一般性，设内部交互为 $T(u,!m)=u'$ 和 $T(v,?m)=v'$，则有 $TG(g,m)=g'$，$g'=\{u',v',senv\}$，g' 中实体的发送消息集合为 $!E'$，g' 中实体的接收消息集合为 $?E'$，$senv$ 的接收消息集合为 $?Eenv$，发送消息集合为 $!Eenv$，

(a) 环境的接收消息集合 $?Eenv$ 必须包含两个实体的发送消息集合之和，表示为 $?Eenv\supseteq !E'\cup !E$；

(b) 如果 g 和 g' 的接收消息集合有重复的元素 $?E'\cap ?E\neq\varnothing$，且 $!E'\neq\varnothing$，那么 $!Eenv$ 可以是 $?E'$ 和 $?E$ 之交的任意一个子集，表示为 $!Eenv\subseteq !E'\cap ?E$；

(c) 如果 g 和 g' 的接收消息集合有重复的元素 $?E'\cap ?E\neq\varnothing$，且 $!E'=\varnothing$，那么 $!Eenv$ 可以是 $?E'$ 和 $?E$ 之交的任意一个非空子集，表示为 $!Eenv\subseteq !E'\cap ?E$ 且 $!Eenv\neq\varnothing$；

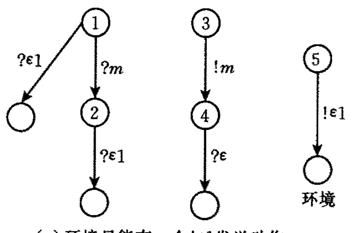
(d) 如果 g 和 g' 的接收消息集合没有重复的元素 $?E'\cap ?E=\varnothing$，那么 $!Eenv$ 为空，表示为 $!Eenv=\varnothing$．显然，如果 $!E'=\varnothing$，则 g' 中不存在可执行的内部交互时，g' 属于非法状态，因为 g' 中只存在接收动作，自动机无法继续前进，表示接口交互存在不一致行为；

(e) 前面对 g 和 g' 的分析是传递的；

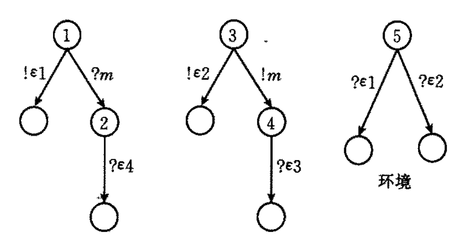
(2) 对于其它状态 g，环境的接收消息集合 $?Eenv$ 包含 g 中实体的发送消息集合，表示为 $?Eenv\supseteq !E$；如果 $!E$ 为空，则环境的发送消息集合 $!Eenv$ 可以是 $?E$ 的任意一个非空子集，表示为 $!E=\varnothing\Rightarrow !Eenv\subseteq ?E\wedge !Eenv\neq\varnothing$；否则 $!Eenv$ 是 $?E$ 的任一子集，即 $!E\neq\varnothing\Rightarrow !Eenv\subseteq ?E$．

定义 3(1)中的(b)和(c)的区别是：表示当 g 和 g' 中实体的接收消息集合有重复的元素时，如果实体的发送消息集合不为空，那么环境的发送消息可以为空；否则环境的发送消息不能为空（如果实体和环境的发送消息集都为空，g' 就因为没有可以执行的动作而成为死锁状态）．需要注意，定义中第一类情况要求对 g 和 g' 的分析是传递的：如果 g' 也存在情况(1)，则需要进一步对 g' 进行类似分析，直到不存在(1)所定义的情况为止．这是因为由接口之间交互连接起来的状态都被环境作为一个状态看待，所以环境必须满足所有这些状态的限制条件．

图 4 中例举了环境应该如何满足接口对的限制．图 4(a)中的环境在状态 5 可以有发送消息动作 $!\varepsilon1$，也可以有其它接收消息动作，但是不能有发送消息动作 $!\varepsilon$，因为接口经过 m 消息交互之后，接口 FSM 分别进入状态 2 和 4，虽然这时接口中存在接收动作 $!\varepsilon$，但是它并不存在 m 交互之前的任何一个状态(1 或 3)中．图 4(b)中环境在状态 5 必须存在接收动作 $!\varepsilon1$ 和 $!\varepsilon2$，但是不能有任何发送动作，因此当接口经过内部消息 m 的交互分别到达状态 2 和 4 后，整个协议没有可以执行的动作，说明接口之间存在着固有的行为不一致．



(a) 环境只能有一个!ε1发送动作



(b) 环境必须有?ε1和?ε2，且没有任何发送动作

图 4 环境定义举例

从环境定义上可以看到两点：(1)环境的行为和接口交互的行为是对应的. 合法环境包含确定的和不确定的两种动作，分别对应环境应该满足的限制和环境可以选择的行为. 而所谓对合法环境的定制就是对不确定行为的选择，选择的标准是接口交互是否会出现非法状态，依据则是定义 4 中不确定动作的取值范围. (2)环境的状态和接口交互的状态是对应的，从合法环境中可以看到接口交互实体在任何一个状态中可能执行的动作和必须可执行的动作，而且任意一个组合 $S_A \times S_B$ 的元素对应的环境状态的行为定义是一样的. 因此不失一般性，我们定义：在 *Penv* 中，同一个组合 (s_1, s_2) 对应的 s_{ENV} 是一样的，即任意两个全局状态 (s_1, s_2, s_{ENV}) 和 (s_1', s_2', s_{ENV}')，如果 $s_1 = s_1'$，$s_2 = s_2'$，则 $s_{ENV} = s_{ENV}'$.

定义 4.　合法环境前提下，在任一全局状态 $g(u, v, senv)$ 中，除了接口之间的消息交互外，*Penv* 的可执行动作还包括实体从环境的接收动作 $?Eexe = ?E \cap \max(!Eenv)$ 和实体对环境的发送动作 $!Eexe = !E \cap ?Eenv$.

显然，由定义 3 和定义 4 可知，实体的可执行环境接收动作 $?Eexe$ 总是等于 $!Eenv$ 的最大集 $\max(!Eenv)$，而可执行环境发送动作 $!Eexe$ 就等于自身发送动作 $!E$.

实体的必可执行动作和可选执行动作分别相关于环境的确定动作和不确定动作，可选执行动作是环境可以选择执行的，通过定制环境这部分动作可以不执行，如可执行环境接收动作 $?Eexe$（当 $?Eexe$ 的元素多于一个或存在其它可执行动作时）. 而必可执行动作是环境无法定制或者必须满足的，如内部交互，可执行环境发送动作 $!Eexe$ 和可执行环境接收动作 $?Eexe$ 的元素只有一个且是当前唯一可执行动作时.

定义 5.　合法环境前提下，在 *Penv* 的任一全局状态 $g(u, v, senv)$ 中，实体可选择定义的动作称为状态 g 的可选执行动作，对应于实体的可执行环境接收动作 $?Eexe$（当 $?Eexe$ 的元素大于一个或存在其它可执行动作时）；实体必须定义的动作称为状态 g 的必可执行动作，对应于（任一可达全局状态 g）：

(1) 由内部交互 $T(u, !m) = u'$，$T(v, ?m) = v'$ 引起的全局转移 $Tg(\{s_1, s_2, e\}, m) = \{s_1', s_2', e\}$；

(2) 由接口对环境的发送消息动作 $T(s, !\epsilon) = s'$ 引起的全局转移 $Tg(\{s_1, s_2, e\}, \epsilon) = \{s_1', s_2, e'\}$；

(3) 由 g 的唯一可执行接收动作 $T(s, ?\epsilon) = s'$ 引起的全局转移 $Tg(\{s_1, s_2, e\}, \epsilon) = \{s_1', s_2, e'\}$.

在 *Penv* 中，存在两种非法状态：一种是接口之间存在未指定的接收，另一种是接口之间的内部交互导致环境无法同时满足多个状态的限制条件，最终出现协议自动机不存在可执行的动作而无法继续执行.

定义 6.　在合法环境的前提下，对 *Penv* 的任一全局状态 $g(u, v, senv)$，如果 u 和 v 之间存在未指定的接收，或者 g 没有可执行的动作，则称 g 为非法状态.

在协议 *Penv* 的定义中，我们给出的 *Penv* 状态集合是从初始状态开始通过有限步执行遍历得到的. 现在我们可以更加精确地定义协议 *Penv* 的状态集合：(1)非法状态也许存在其它可执行动作，但是非法状态之后遍历得到的状态是没有意义的；(2)对协议的遍历只能依据每个状态的可执行动作（请注意可选执行动作的定义是和环境相关的），而且每个可执行动作必须搜索到.

定义 7.　在合法环境的前提下，协议 *Penv* 的可达状态搜索依赖于状态的可执行动作，并且搜索过程起始于初始状态，停止于三种状态：非法状态、终止状态和已经遍历过的状态.

既然协议的可达状态集合就是初始状态的可达状态集合，那么可以定义状态的兼容.

定义 8.　如果存在一个合法环境 *ENV*，在接口对和环境 *ENV* 组成的协议 *Penv* 中，使得某个状态 g 的可达状态集合中没有非法状态，且所有的可达状态都能到达终止状态，则称状态 g 是兼容的.

由定义 7 和定义 8 可以得到下面两个引理.

引理 1.　非法状态 g 都是不兼容的.

引理 2.　协议 *Penv* 兼容当且仅当 *Penv* 的初始状态是兼容的.

显然，从兼容状态的定义可知，如果在某个环境 *ENV* 下状态 g 是兼容的，那么 g 的任一可达状态 g' 都是兼容的，因为 g' 的可达状态集合属于 g 的可达状态集合，显然 g' 也满足兼容性要求；反过来，如果在某个合法环境下 g 的所有一步转移得到的状态 g' 是兼容的，那么 g 也是兼容的，因为 g 的可达状态集合等于所有 g' 的可达状态集合之和. 由于任一状态 g 的转移动作分为必可执行动作和可选执行动作，对于可选执行动作得到的状态 g'，如果 g'' 是不兼容的，那么可以通过环境行为的选择，使 g'' 成为不可达状态；对于必可执行动作得到的状态 g''，如果 g'' 不兼容，则 g 必然是不兼容的. 由此可得如下定义.

定义 9.　对于协议 *Penv* 中的状态 g 和经过一步转移 T' 得到的状态 g'，当满足下面两个条件时，g

是不兼容的：

(1) 如果 g' 是不兼容的，且 T' 是必可执行动作时，那么 g 也是不兼容的；

(2) 如果所有的 T' 都是可选执行动作，且所有的 g' 都不兼容，则 g 也是不兼容的.

由引理1、引理2和定义9可以很容易想到计算协议 *Penv* 兼容性的办法，就是从 *Penv* 中剔除所有不兼容的状态，当最后留下一个包含初始状态的自动机时，则这个自动机就是我们要找的；否则必定得出初始状态是不兼容的，从而得出协议 *Penv* 不可能兼容，即任何合法环境都无法满足 *Penv* 的兼容性. 当我们得到一个自动机时，只要从中删除环境模型的信息，可以轻易的从中得到接口对的组合接口模型.

下面给出划定接口 A,B 兼容的算法 COMP.

算法 1. COMP.

输入：接口 A 和 B 的 FSM 模型

输出：*COMP*(A,B)

1. 根据接口 A、B 和定义5构建 *Penv*.

2. 令集合 Q 为状态集合，初始化 Q 为 *Penv* 中的非法状态集合.

3. 从 Q 中取一个元素 g.

4. 对于 *Penv* 中每个存在转移 $Tg(g',m)=g$ 的状态 g'，如果 g' 满足下面两个条件，则将 g' 标记为不兼容状态并放入 Q 中.

(a) $Tg(g',m)=g$ 是 g' 的必可执行动作；

(b) 在当前 *Penv* 中 $Tg(g',m)=g$ 是 g' 剩下的唯一可执行动作.

5. 如果步4中所有状态 g' 处理完毕，将 g 从 *Penv*(同时包含所有相关的转移)和 Q 中删除.

6. 跳到步3继续执行，直到 Q 为空.

7. 返回结果 *Penv*.

算法的核心思想就是从非法状态开始，逐个地找出并剔除不兼容的状态，如果最后初始状态也被剔除了，那么说明接口 A 和 B 是不兼容的；否则必定存在一个包含初始状态和终止状态的有限状态自动机 *Penv'*，自动机中所有的状态都是兼容的. 为了进一步剔除冗余状态，需要对结果 *Penv* 进一步计算可达状态集合，去掉那些不可达的状态. 由此可以看到，通过 COMP 算法得到的组合自动机模型是最大可能的组合自动机，因为它包含所有可能使接口对兼容的行为特性，即包含了所有可能的兼容状态，它得到的是最大兼容状态集合，添加任意一个其它状态，要么会使新的自动机不兼容，要么这个新加的状态是不可达的.

COMP 算法得到的 *Penv* 是包括环境接口信息的协议 FSM 模型，为了得到接口对的组合模型，需要将环境信息从 *Penv* 中删掉. *Penv* 的任意一个状态都是组合模型和环境状态组成，由于已经设定任意两个全局状态 (s_1,s_2,s_{ENV}) 和 (s'_1,s'_2,s'_{ENV})，如果 $s_1=s'_1$，$s_2=s'_2$，则 $s_{ENV}=s'_{ENV}$，那么可以得出 *Penv* 中任意一个组合 (s_1,s_2) 只可能出现一次，由此可以从 *Penv* 中得到组合接口模型的状态集合；另外，将 *Penv* 中的所有动作的环境信息去掉，并添加消息的方向，就能得到组合接口模型的转移 T.

对 COMP 算法得到的 *Penv*，通过下面两个处理，可以得到接口对的组合接口模型.

(1) 将任意一个状态 $(s_1,s_2,s_{\mathrm{ENV}})$ 去除环境状态，变为 (s_1,s_2).

(2) 对任意一个全局转移 $Tg(\{s_1,s_2,e\},m)=\{s'_1,s'_2,e'\}$

(i) 如果是接口对环境的发送消息动作如 $T(s_1,!\varepsilon)=s'_1$，则改为 $Tg(\{s_1,s_2\},!\varepsilon)=\{s'_1,s_2\}$；

(ii) 如果是接口从环境的接收消息动作如 $T(s_1,?\varepsilon)=s'_1$，则改为 $Tg(\{s_1,s_2\},?\varepsilon)=\{s'_1,s_2\}$；

(iii) 如果是内部交互 $T(u,!m)=u'$，$T(v,?m)=v'$，则改为 $Tg(\{s_1,s_2\},m)=\{s'_1,s'_2\}$.

将上述的内容添加到 COMP 算法中，可以使 COMP 算法得到一个完整的组合接口模型. 当存在多个接口的时候，可以用相同的计算来判断 N 个接口是否是兼容的，其根据就是每次 COMP 算法的结果也是一个合法的接口自动机，那么就可以作为 COMP 算法的输入，判断和其它接口的兼容性. 在 COMP 算法检查时，如果两个接口都不存在与环境的交互，那么得到的算法结果就是一个完整的协议，表示 N 个接口是兼容的，而且这个协议就是 N 个接口的交互协议.

COMP 算法和文献[9]的接口自动机具有相同的时间复杂度，为了满足会话类 E-Service 的会话完整性，需要对 *Penv* 中的状态做更多的判断和裁剪，但是这个操作不会增加复杂度，因为整个 *Penv* 的复杂度是有限的.

引理 3. 给定两个会话类 E-Service 的接口 FSM 模型 A 和 B，可以用 COMP 算法判定它们是否接口兼容，同时得到组合自动机模型，其时间复杂度与接口规模 $|A|$ 和 $|B|$ 呈线性关系.

6 实例验证

现在我们就把算法应用到图3中的旅游例子，旅游例子由 TravelAgent，Airline 和 Traveler 三个

自动机组成，我们验证的方法就是对三个自动机进行逐个的接口兼容算法，直到最后得到一个完整的协议，表示这三个自动机是兼容的，而且 *Penv* 就是它们的交互协议.

图 3(a)子图表示的是旅游代理和飞机票代理接口组合时得到的组合自动机 *Penv*，其中状态 8 是非法状态. 那么根据算法，状态 8 将放入集合 Q 中，分析状态 8 的父状态 1，状态 1 是从环境接收 *rsvCanReq* 消息后才到转移到 8，而 ?*rsvCanReq* 不是状态 1 的唯一可执行动作，那么状态 1 尚不能判定为不兼容状态. 但是状态 1 是 8 的唯一的父状态，因此可以把状态 8 从 Q 中删除.

由于状态 8 是整个组合自动机中唯一的非法状态，Q 中不再有任何其它元素，那么算法结束，由此得到了 TravelAgent 和 Airline 的组合自动机 *COMP*(TravelAgent, Airline)，也就是图 3(a)中删除了状态 8 及其相关转移的结果.

接下来，再对组合自动机 *COMP*(TravelAgent, Airline)和 Traveler 应用算法，由此可以得到 *COMP*(*COMP*(TravelAgent, Airline), Traveler)，由于已经没有其它的自动机，而且 *COMP*(TravelAgent, Airline)和 Traveler 的交互中也不存在与环境的交互，因此可以确定最后得到的就是旅游例子的完整协议模型 *Penv*，而这个协议模型正好是图 3(c)，从而简单地验证了算法的正确性.

7 结　论

形如 $COMP(COMP(A_1,\cdots,COMP(A_{N-1},A_N)))$ 的计算一般不会直接用来寻找兼容协议，因为从全体会话类服务集合中进行计算，其计算量是非常大的. 一般是和其它自动服务组合技术结合使用，特别是作为一种必要的辅助技术，在会话类服务的查找和组合过程中，提供额外的更强的检查和验证，以提高查找和组合的效率和成功率.

在具体方法的实现中，还有些问题需要解决. 如 Plan(规划)等方法都是以服务方法为粒度来建模(为 Action)，通过对 Action 的顺序排列来满足某个目标，这和本文的对整个服务而不是单个方法的建模方法是不一致的，服务在我们的工作中被建模为一个完整的模型，模型中的操作不能随意地调用(除了状态无关的信息查询类操作)，操作调用序列需要满足一定的逻辑关系，从服务中任意选取的一个操作不能保证其调用会成功，所以不能将本文的接口兼容直接应用在 Plan 或推理类的方法中. 为了将本文的结果应用于动态组合方法，需要对当前可行的动态组合方法做一定的修改.

由于 E-Service 是一个自治的独立系统，其实现情况在现实系统中有多种方式，同步环境下的行为只是它的可能行为特征的一部分，如果考虑通信环境、实现中的队列等因素的影响，那么还需要考虑异步环境的行为和兼容，这样才能完整地表现 E-Service 的行为特征. 因此下一步的工作可以对异步环境下的 E-Service 交互行为进行分析和验证.

参 考 文 献

1 Andrews T., Curbera F., Dholakia H.. Business Process Execution Language for Web Services (Version 1.0). IBM, 2003
2 Brand D., Zafiropulo P.. On communicating finite-state machines. Journal of the ACM, 1983, 30(2): 323～342
3 Christophides V., Hull R., Karvounarakis G.. Beyond discrete e-services: Composing session-oriented services in telecommunications. In: Proceedings of the Workshop on Technologies for E-Services (TES), Rome, Italy, 2001, 58～60
4 Hopcroft J. E., Ullman J. D.. Introduction to Automata Theory, Languages, and Computation, 2nd Edition. Boston, MA, USA: Addison-Wesley, 2001.
5 McIlraith S. A., Son T. C., Zeng H.. Semantic Web services. IEEE Intelligent Systems, 2001, 16(2): 50～63
6 Box D., Ehnebuske D., Kakivaya G.. Simple Object Access Protocol (soap) 1.1. W3C Note, 2000
7 Arkin A., Askary S., Fordin S.. Web Service Choreography Interface (WSCI) 1.0. W3C Note, 2002
8 Christensen E., Curbera S., Meredith G.. Web Services Description Language (WSDL) 1.1. W3C Note 15, March 2001
9 Alfaro L., Henzinger T.. Interface automata. ACM SIGSOFT Software Engineering Note, 2001, 26(5): 109～120
10 Bultan T., Fu X., Hull R.. Conversation specification: A new approach to design and analysis of e-service composition. In: Proceedings of the 12th International World Wide Web Conference, Budapest, Hungary, 2003, 403～410
11 Solanki M., Cau A., Zedan H.. Augmenting semantic Web service descriptions with compositional specification. In: Proceedings of the WWW, ACM, Manhattan, NY, USA, 2004, 544～552
12 Beyer D., Chakrabarti A., Henzinger T. A.. Web service interfaces. In: Proceedings of the WWW, ACM, Manhattan, NY, USA, 2004, 148～159
13 Yellin D. M., Robert Strom E.. Protocol specifications and component adaptors. In: Proceedings of the ACM Transactions on Programming Languages and Systems (TOPLAS), New York, 1997, 292～333
14 Foster H., Uchitel S., Magee J.. Compatibility verification for Web service choreography. In: Proceedings of the ICWS, San Diego, California, 2004, 738～741

15 McIlraith S., Son T. C.. Adapting golog for composition of semantic Web services. In: Proceedings of the International Conference Principles of Knowledge Representation and Reasoning (KRR 02), Toulouse, France, 2002, 482～493

16 Martin D., McIlraith S., Bringing semantics to Web services: The OWL-S approach. IEEE Intelligent System, 2003, 18 (1): 90～93

17 Wu D., Parsia B., Sirin E.. Automating DAML-S Web services composition using SHOP2. In: Proceedings of the 2nd International Semantic Web Conference (ISWC2003), Sanibel Island, Florida, USA, 2003, 195～210

18 McDermott D.. Estimated-regression planning for interaction with Web services. In: Proceedings of the 6th International Conference on AI Planning and Scheduling, Toulouse, France, 2002, 204～211

19 Harel D.. STATECHARTS: A visual formalism for complex systems. Science of Computer Programming, 1987, 8(3): 231～274

20 Fu X., Bultan T., Su J. W.. Conversation protocols: A formalism for specification and verification of reactive electronic services, theoretical computer science (TCS). Special Issue on Selected Papers from the 8th International Conference on Implementation and Application of Automata (CIAA 2003), Santa Barbara, CA, USA, 2004, 328(1/2): 19～37

21 Betin-Can A., Bultan T., Fu X.. Design for verification for asynchronously communicating Web services. In: Proceedings of the 14th International World Wide Web Conference, Chiba, Japan, 2005, 750～759

22 Fu X., Bultan T., Su J. W.. Analysis of interacting BPEL Web services. In: Proceedings of the 13th International World Wide Web Conference (WWW 2004), New York, USA, 2004, 621～630

23 Zhang W. T., Peng Y., Chen J. L.. Investigation on interface compatibility of session-orient E-Service. Journal of Beijing University of Posts and Telecommunications, 2006, 29(supplement): 140～143



ZHANG Wen-Tao, born in 1979, Ph. D. candidate. His research interests include E-Service formalism and semantic description, service discovery and (dynamic) composition, protocol verification.

PENG Yong, born in 1978, Ph. D., lecturer. His research interests include network service, mobile network.

CHEN Jun-Liang, born in 1933, professor and Ph. D. supervisor, member of Chinese Academy of Sciences and Chinese Academy of Engineering. His research interests include communication network, communication software.

Background

The common service composition approach, especially dynamic composition, prefers to the function composition. Function composition depicts single service method as an action. Through matching the input/output and/or the precondition/effects of the action, a series of actions is connected and the expected goal function is included in these actions. The support technologies and standards include WSDL, OWL-S, and AI (plan, Golog etc.).

Session-oriented E-Service's observable behavior presents a defined choreography of messages, characterized in terms of temporal and logical dependencies among the exchanged messages, and the message sequence has a "start" and an "end". Hence, the composition of Session-oriented E-Service is also featured as a protocol of interaction. Sequentially the static interface description languages such as WSDL are not sufficient to depict the behavior interface of Session-oriented E-Service, and function composition can not ensure the composition works. This paper presents an automata based formalism model of E-Service interface to capture the temporal aspect of message sequence, investigates interface compatibility check for design-time verification of service composition, and provides an algorithm to obtain the legal composition interface model, as well as the interface composition check. Interface compatibility check provides design-time verification of service composing which can assist (dynamic) composer to ensure the correctness of composition in term of illegal state and deadlock or no termination (named protocol compatible).